Béziers for Beginners

Bézier curves are parametric curves defined by a set of control points. They are based on the polynomials developed by Sergei Natanovich Bernstein in 1912.

In 1959 Paul de Casteljau developed de Casteljau's algorithm, to create the curves and applied them to computer-aided design at French automaker Citroën. De Casteljau's method was not published until the 1980s. In the 1960s, Pierre Bézier, discovered the method independently and used them to design auto bodies at Renault a competing french automobile company. Bézier's method was widely popularized and the curves bear his name to this day.

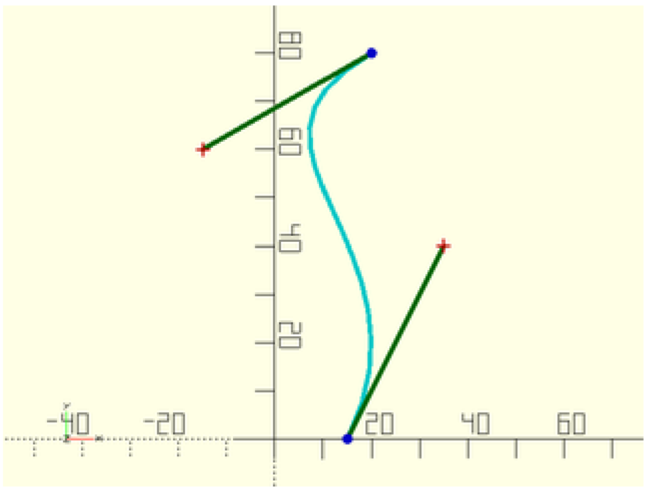
The positions of the Bézier control points in relation to one another define the shape of the curve. The simplest cubic Bézier curves have 4 control points. The first and last control points are the endpoints of the curve. The curve starts in the direction of the second control point and then turns so that it arrives at the endpoint from the direction of the third control point.

To work with Béziers in OpenSCAD we need to load the Bézier extension BOSL2/beziers.scad in addition to BOSL2/std.scad.

In OpenSCAD the control points of a Bézier curve are contained in a list.

To visualize a Bézier curve we can use the module debug_bezier().

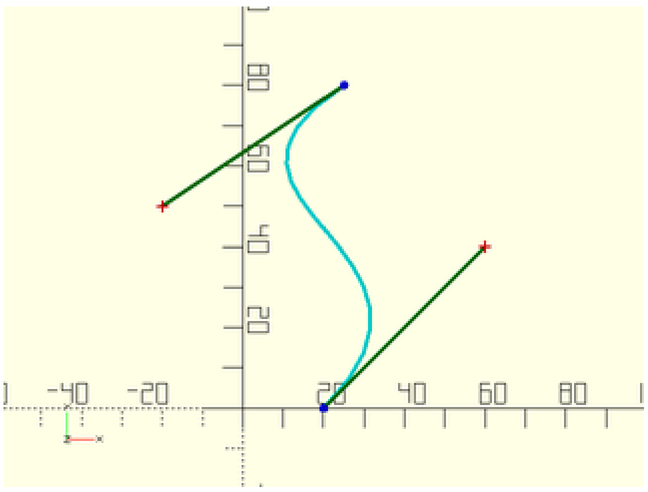
```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

bez = [[15,0], [35,40], [-15,60], [20,80]];
debug_bezier(bez);
```



By moving the second and third points on the list we change the shape of the curve.

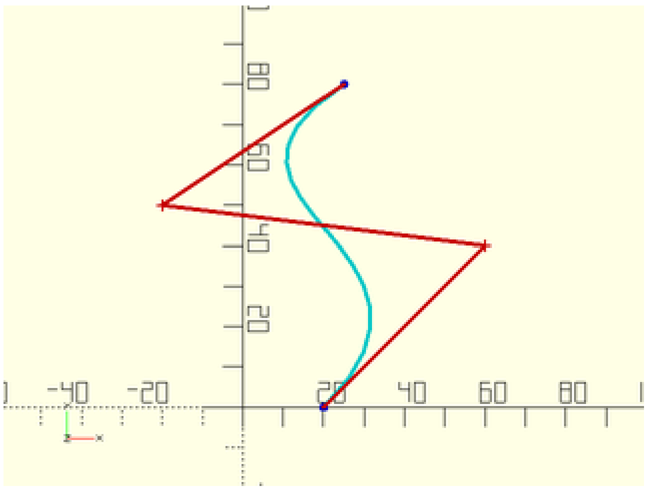
```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>
```

```
bez = [[20,0], [60,40], [-20,50], [25,80]];
debug_bezier(bez);
```



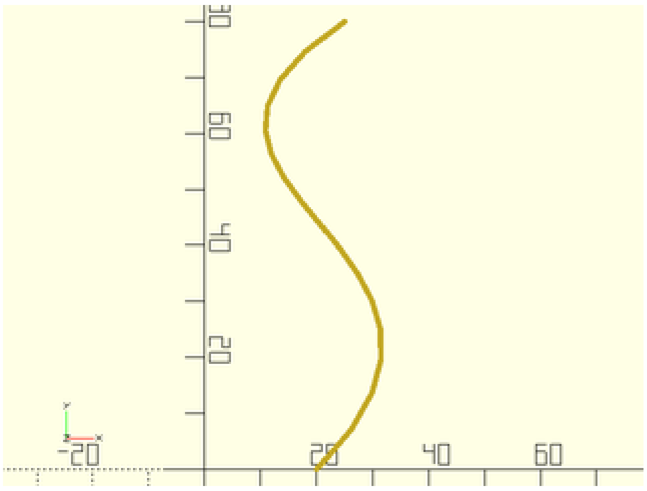
While bez in our example is a list of points, it is not the same as OpenScad path, which is also a list of points.

```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

bez = [[20,0], [60,40], [-20,50], [25,80]];
debug_bezier(bez);
color("red") stroke(bez);
```



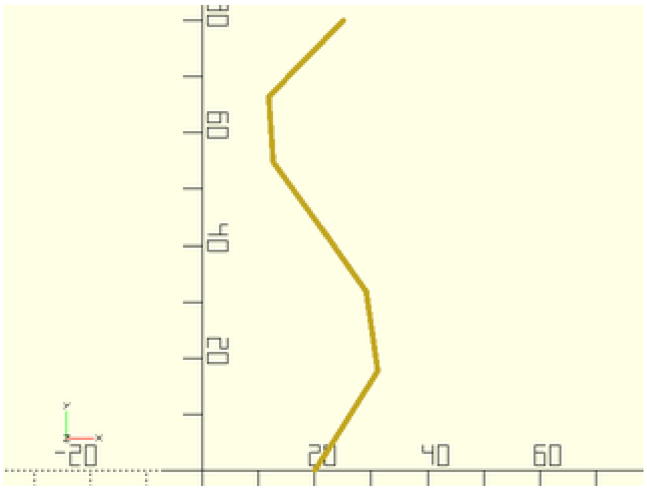
We can use the bezpath_curve() function to convert the Bézier curve to an OpenSCAD path.

```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

bez = [[20,0], [60,40], [-20,50], [25,80]];
path = bezpath_curve(bez);
stroke(path);
```



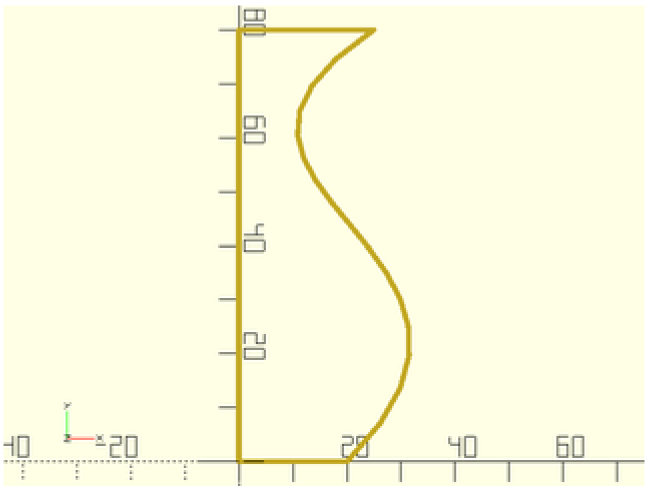
By default bezpath_curve() splits the Bézier curve into 16 straight-line segments. Note that the special variable $FN has no effect on the number of steps. You can control this number using the splinesteps argument.

```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

bez = [[20,0], [60,40], [-20,50], [25,80]];
path = bezpath_curve(bez, splinesteps = 6);
stroke(path);
```



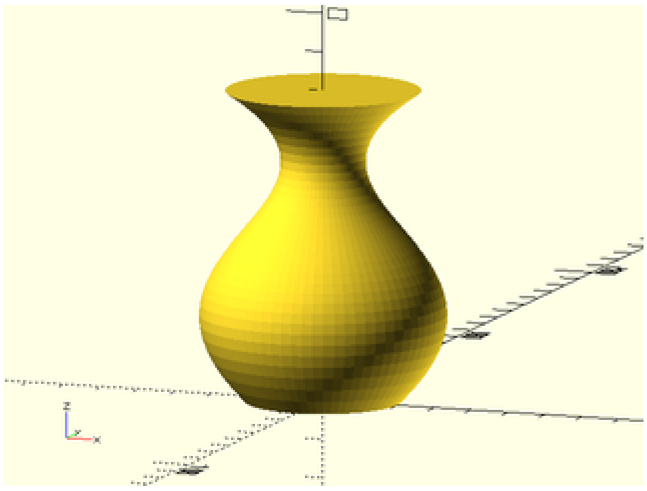
To close the path to the x-axis we can use the bezpath_close_to_axis() function.

```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

bez = [[20,0], [60,40], [-20,50], [25,80]];
closed = bezpath_close_to_axis(bez, axis = "Y");
path = bezpath_curve(closed);
stroke(path);
```



If we use rotate_sweep() to sweep that path around the y-axis we have a solid vase-shaped object. Here we're using both $fn and the splinesteps argument to produce a smoother object.

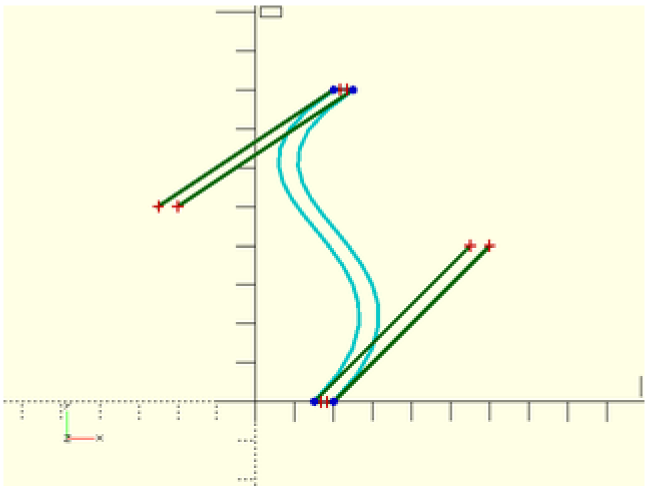
```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>
$fn = 72;

bez = [[20,0], [60,40], [-20,50], [25,80]];
closed = bezpath_close_to_axis(bez, axis = "Y");
path = bezpath_curve(closed, splinesteps = 32);
rotate_sweep(path,360);
```



Instead of closing the path all the way to the y-axis, we can use bezpath_offset() to duplicate the path 5 units to the left, and close the two paths at the top and bottom. Note that bezpath_offset takes an x,y pair as an offset value.

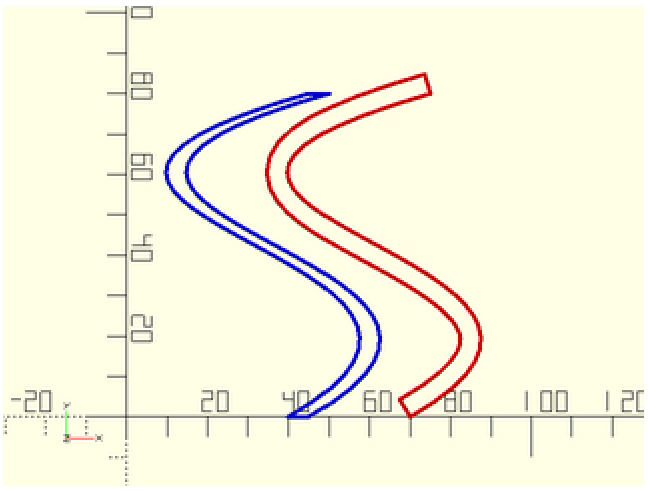
```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>
$fn = 72;

bez = [[20,0], [60,40], [-20,50], [25,80]];
```

```
closed = bezpath_offset([-5,0], bez);
debug_bezier(closed);
```



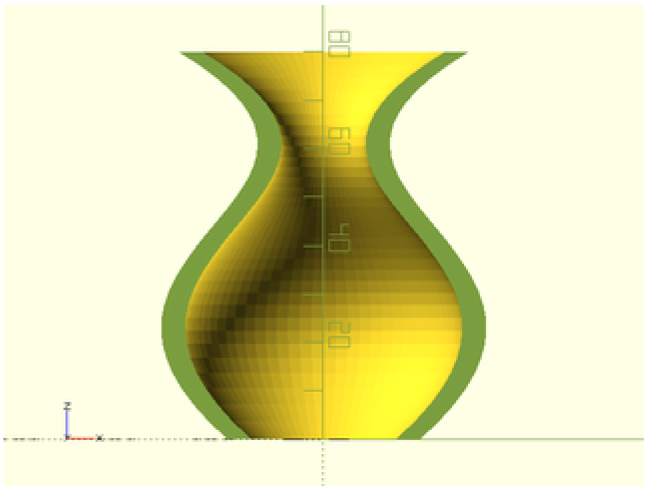
Note that bezpath_offset() does not ensure a uniform wall thickness. For a constant-width wall we need to offset the path along the normals. This we can do using offset(), but we must first convert the Bézier to an OpenSCAD path, then reverse the offset path to create a closed path. You can see the difference between the two methods here, with bezpath_offset() in blue, and offset() in red.

```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>
$fn = 72;

bez = [[40,0], [110,40], [-60,50], [45,80]];

bez2 = bezpath_offset([5,0], bez);
closed= bezpath_curve(bez2, splinesteps = 32);
color("blue") stroke(closed);

path2 = bezier_curve(bez, splinesteps = 32);
closed2 = concat(path2,reverse(offset(path2,delta=5)),[bez[0]]);
right(30) color("red") stroke(closed2);
```



Sweeping a Bézier path offset using either method around the y-axis gives us a shape with an open interior. However, as this cross section shows, our new path does not close the bottom of the vase.

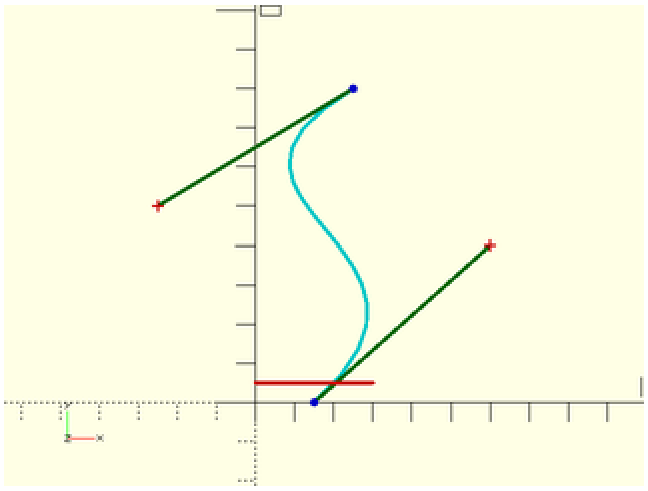
```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

$fn = 72;

bez = [[15,0], [60,40], [-25,50], [25,80]];
closed = bezpath_offset([5,0], bez);
path = bezpath_curve(closed, splinesteps = 32);
back_half(s = 200) rotate_sweep(path,360);
```



We'll use a cylinder with a height of 5 for the floor of our vase. At the bottom of the vase the radius of the hole is bez[0].x but we need to find the inside radius at y = 5. The function bezier_line_intersection() will return a list of u-values where a given line intersects our Bézier curve. The function bezier_points() will convert that list of u-values to a list of x,y coordinates. Drawing a line at y = 5 gives us the single-element list [[20.1072,5]].

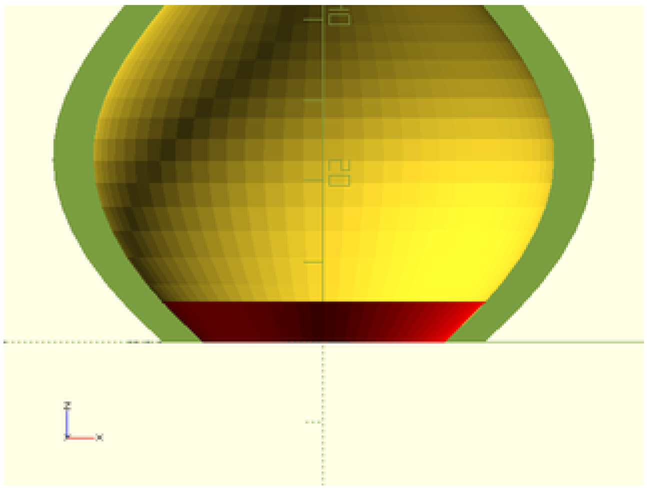
```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>
```

```
bez = [[15,0], [60,40], [-25,50], [25,80]];
debug_bezier(bez);
line = [[0,5], [30,5]];
color("red") stroke(line);
u = bezier_line_intersection(bez,line);
echo(bezier_points(bez,u));  //   [[20.1072,5]]
```



That means a cyl() with a height of 5, a bottom radius of bez[0].x and a top radius of 20.1072 will just fit our vase.

```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

$fn = 72;

bez = [[15,0], [60,40], [-25,50], [25,80]];
closed = bezpath_offset([5,0], bez);
path = bezpath_curve(closed, splinesteps = 32);
back_half(s = 200) rotate_sweep(path,360);
color("red") cyl(h = 5, r1 = bez[0].x, r2 = 20.1072, anchor = BOT);
```



Keep in mind the fact that **$fn** controls the "roundness" of OpenSCAD objects other than Bézier curves. Bézier's smoothness is controlled by the **splinesteps** argument.

```
include<BOSL2/std.scad>
include<BOSL2/beziers.scad>

$fn = 72;

bez = [[15,0], [40,40], [-20,50], [20,80]];
closed = bezpath_offset([2,0], bez);
path = bezpath_curve(closed, splinesteps = 64);

rotate_sweep(path,360, $fn = 72);
right(60) rotate_sweep(path,360, $fn = 6);
right(120) rotate_sweep(path,360, $fn = 4);
```